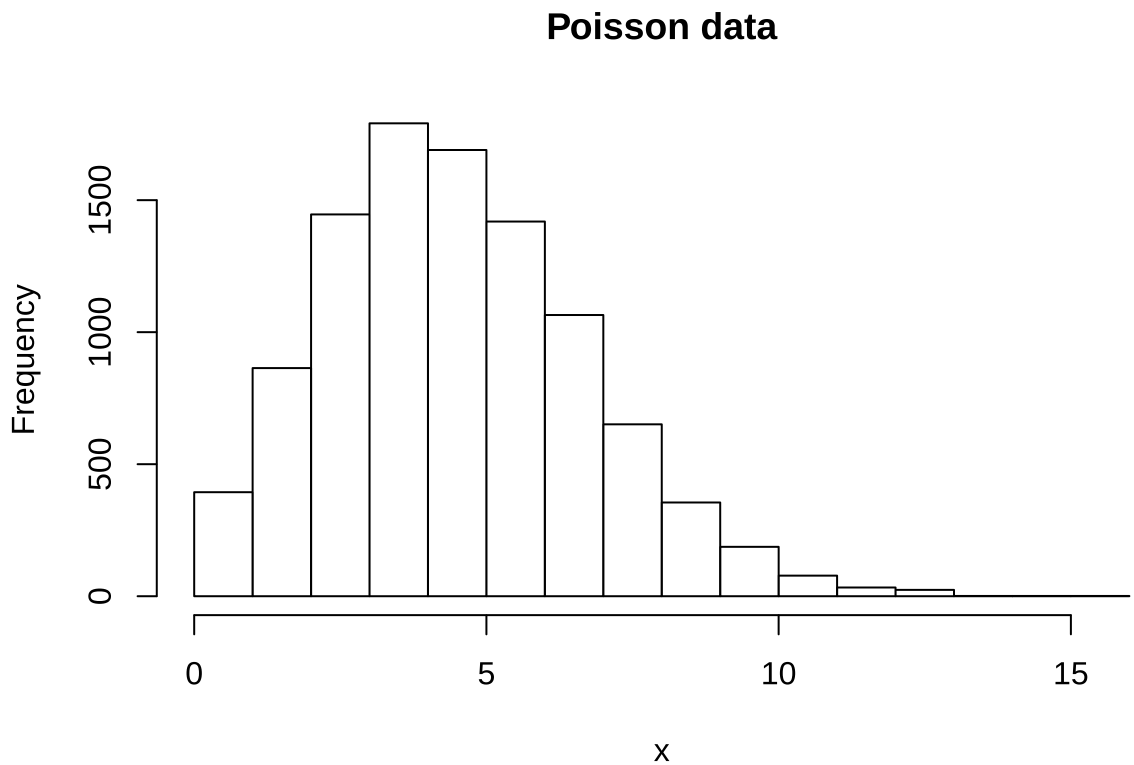
# Call Center Data

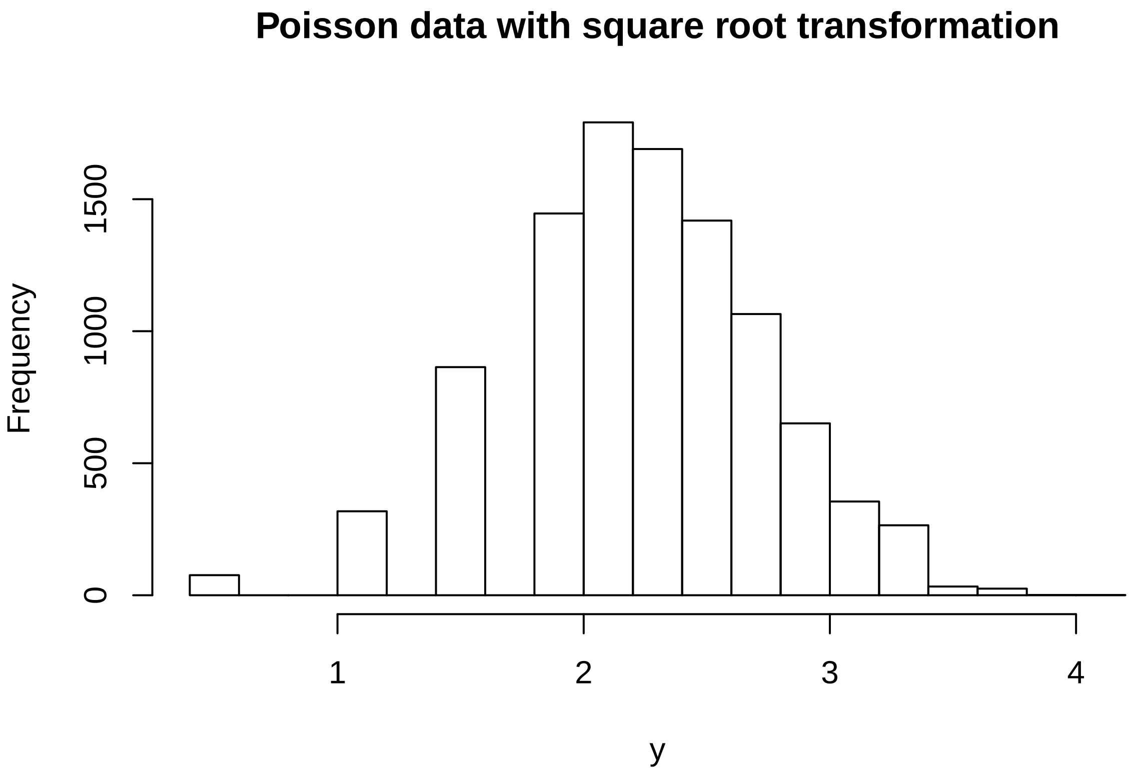
*Syed Rahman*

*2/6/2017*

### Call Center Data Analysis

The goal is to predict the number of calls coming in during the second half of the day using the number of calls recieved during the first half of the day. To this end suppose $x_1$ and $x_2$ is the number of calls received during the first half of the day and the second half of the day, respectively. Let $x = (x_1^t, x_2^t)^t$ and $y = \sqrt{x + \frac{1}{4}}$. It can be shown that if $x \sim Poisson(\lambda)$, then $y \dot{\sim} \mathcal{N}(\mu, \Sigma)$.






Suppose we partition the covariance matrix of $y$ as follows:

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$$

In such a case, $y_2|y_1 \sim \mathcal{N}(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(y_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})$. Hence the best mean squared predictor of $y_2$ given $y_1$ (i.e. the preidctor that minimizes mean squared error) is

$$\mathbb{E}[y_2|y_1] = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(y_1 - \mu_1).$$

This is done using the follow code:

```
predict.mean <- function(x1, mu, Sigma) {
    p1 <- length(x1)
    p <- length(mu)
    p2 <- p - p1
    mu1 <- mu[1:p1]
    mu2 <- mu[(p1 + 1):p]
    Sigma11 <- Sigma[1:p1, 1:p1]
    # Sigma12 <- Sigma[1:p1,(p1+1):p]
    Sigma21 <- Sigma[(p1 + 1):p, 1:p1]
    # Sigma22 <- Sigma[(p1+1):p,(p1+1):p]
    x2 <- mu2 + Sigma21 %*% solve(Sigma11, x1 - mu1)
    return(x2)
}
```

Hence we must estimate $\mu_2$ and $\Sigma$ from the data. To this end, we divide the dataset into a training datset and a test dataset. Typically we do this in a random fashion, but since this is time-series data, I take the first 70% of the data to be the training dataset and the remaining 30% of the dataset to be the test dataset. Once we have the estimates, we make predictions for the calls coming in for the second half of the day using the calls that came in for the first half of the day for the test data set and then compare with true values.

The data was obtained from http://iew3.technion.ac.il/serveng/callcenterdata, which contains daily data for a call center for an Israeli bank for the year 1999. Four days are missing. But for all the other 361 days we have information for time of call, length of call, etc. As the call center was only open from 7AM to midnight, I removed all the data points outside of this interval.

```
months = format(ISOdate(1999, 1:12, 1), "%B")
months = tolower(months)
data = read.table("january.txt", header = TRUE)
for (month in months[-1]) {
    data = rbind(data, read.table(paste(month, ".txt", sep = ""),
        header = TRUE))
}
head(data, n = 6)
```

```
##   vru.line call_id customer_id priority type   date vru_entry vru_exit
## 1   AA0101   33116     9664491        2   PS 990101   0:00:31  0:00:36
## 2   AA0101   33117           0        0   PS 990101   0:34:12  0:34:23
## 3   AA0101   33118    27997683        2   PS 990101   6:55:20  6:55:26
## 4   AA0101   33119           0        0   PS 990101   7:41:16  7:41:26
## 5   AA0101   33120           0        0   PS 990101   8:03:14  8:03:24
## 6   AA0101   33121           0        0   PS 990101   8:18:42  8:18:51
##   vru_time q_start  q_exit q_time outcome ser_start ser_exit ser_time
## 1        5 0:00:36 0:03:09    153    HANG   0:00:00  0:00:00        0
## 2       11 0:00:00 0:00:00      0    HANG   0:00:00  0:00:00        0
## 3        6 6:55:26 6:55:43     17   AGENT   6:55:43  6:56:37       54
## 4       10 0:00:00 0:00:00      0   AGENT   7:41:25  7:44:53      208
## 5       10 0:00:00 0:00:00      0   AGENT   8:03:23  8:05:10      107
## 6        9 0:00:00 0:00:00      0   AGENT   8:18:50  8:23:25      275
##      server
## 1 NO_SERVER
## 2 NO_SERVER
## 3    MICHAL
## 4     BASCH
## 5    MICHAL
## 6     KAZAV
```

I divide the day up into 10 minute chunks and the first goal is to reformat the data so that there is an easy way to view the number of calls received at the call center on each day for each time period.

```
data = filter(data, outcome == "AGENT")  ### keeping only agent outcome, rest are hangups or transforms
data = select(data, date, vru_entry)  ### only interested in date and vru entry. disccarding all other 
data$timedate = paste(data$date, data$vru_entry)
data$timedate = strptime(data$timedate, format = "%y%m%d %H:%M:%S")  ### converting to R date format
tt = seq(from = ISOdate(1999, 1, 1, 0, 0, 0, tz = "EST"), to = ISOdate(1999,
    12, 31, 0, 0, 0, tz = "EST"), by = "10 min")  ### creating 10 minute chunks
data$timeperiods = cut(data$timedate, breaks = tt)  ### dividing up calls accoridn to 10 minute chunks

data$periods <- sapply(strsplit(as.character(data$timeperiods),
    " "), "[", 2)  ### keeping only the time period. date discarded

data$timeperiods <- as.character(data$timeperiods)
data$count <- as.numeric(ave(data$timeperiods, data$timeperiods,
    FUN = length))  ### creating a column with counts of how many calls were received in a certain time

data2 = select(data, date, periods, count)  ### keeping only relevant columns such as date, periods and
```

```r
data_wide <- reshape(data2, timevar = "periods", idvar = c("date"),
    direction = "wide")

data_wide[is.na(data_wide)] = 0

### reshaping long format to a wide format and setting all NA
### values to 0

#### reordering column by time
data_wide = data_wide[, order(names(data_wide))]
```

After all the preprocessing the data in wide format look something like this:

```r
head(data_wide, n = 3)
```

```
##    count.00:00:00 count.00:30:00 count.00:40:00 count.00:50:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36              0              0              0              0
##    count.01:00:00 count.01:10:00 count.01:20:00 count.01:40:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36              0              0              0              0
##    count.01:50:00 count.02:30:00 count.03:30:00 count.03:40:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36              0              0              0              0
##    count.04:10:00 count.04:20:00 count.04:30:00 count.04:40:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36              0              0              0              0
##    count.04:50:00 count.05:00:00 count.05:10:00 count.05:20:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36              0              0              0              0
##    count.05:30:00 count.05:40:00 count.05:50:00 count.06:00:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36              0              0              0              0
##    count.06:10:00 count.06:20:00 count.06:30:00 count.06:40:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36              0              0              0              0
##    count.06:50:00 count.07:00:00 count.07:10:00 count.07:20:00
## 1               1              6              3              4
## 26              0              0              0              0
## 36              3              8             13             12
##    count.07:30:00 count.07:40:00 count.07:50:00 count.08:00:00
## 1               8             10              9              9
## 26              0              0              0              0
## 36             13             10              8             12
##    count.08:10:00 count.08:20:00 count.08:30:00 count.08:40:00
## 1              15             24             10             17
## 26              0              0              0              0
```

```
## 36             21             23             20             27
##    count.08:50:00 count.09:00:00 count.09:10:00 count.09:20:00
## 1              16             15             16             17
## 26              0              0              0              0
## 36             26             21             21             14
##    count.09:30:00 count.09:40:00 count.09:50:00 count.10:00:00
## 1              12             24             11             14
## 26              0              0              0              0
## 36             19             23             31             26
##    count.10:10:00 count.10:20:00 count.10:30:00 count.10:40:00
## 1              15             17             10             17
## 26              0              0              0              0
## 36             33             23             29             23
##    count.10:50:00 count.11:00:00 count.11:10:00 count.11:20:00
## 1               8             16             13             13
## 26              0              0              0              0
## 36             21             26             24             19
##    count.11:30:00 count.11:40:00 count.11:50:00 count.12:00:00
## 1              14             18             12              8
## 26              0              0              0              0
## 36             26             21             16             25
##    count.12:10:00 count.12:20:00 count.12:30:00 count.12:40:00
## 1              11             11              7             10
## 26              0              0              0              0
## 36             23             18             18             20
##    count.12:50:00 count.13:00:00 count.13:10:00 count.13:20:00
## 1               9              7              5             10
## 26              0              0              0              0
## 36             21             16             28             22
##    count.13:30:00 count.13:40:00 count.13:50:00 count.14:00:00
## 1              11             11             11              0
## 26              0              0              0              0
## 36             21             18             27             25
##    count.14:10:00 count.14:20:00 count.14:30:00 count.14:40:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36             20             24             27             23
##    count.14:50:00 count.15:00:00 count.15:10:00 count.15:20:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36             23             25             27             31
##    count.15:30:00 count.15:40:00 count.15:50:00 count.16:00:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36             23             19             25             18
##    count.16:10:00 count.16:20:00 count.16:30:00 count.16:40:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36             32             32             33             39
##    count.16:50:00 count.17:00:00 count.17:10:00 count.17:20:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36             29             33             27             30
##    count.17:30:00 count.17:40:00 count.17:50:00 count.18:00:00
```

```
## 1               0              0              0              0
## 26              0              0              0              0
## 36             21             23             14             21
##    count.18:10:00 count.18:20:00 count.18:30:00 count.18:40:00
## 1               0              0              0              0
## 26              0              0              0              0
## 36             12             14              9             12
##    count.18:50:00 count.19:00:00 count.19:10:00 count.19:20:00
## 1               0              0              0              0
## 26              3             17             12              7
## 36             15              4             12             16
##    count.19:30:00 count.19:40:00 count.19:50:00 count.20:00:00
## 1               0              0              0              0
## 26              7             12             12              7
## 36             16             15             12             16
##    count.20:10:00 count.20:20:00 count.20:30:00 count.20:40:00
## 1               0              0              0              0
## 26              4              6              9              6
## 36             10             12             14             12
##    count.20:50:00 count.21:00:00 count.21:10:00 count.21:20:00
## 1               0              0              0              0
## 26              8              7              9              9
## 36             11              7             10              5
##    count.21:30:00 count.21:40:00 count.21:50:00 count.22:00:00
## 1               0              0              0              0
## 26              7              7              4              9
## 36             13             15             12              8
##    count.22:10:00 count.22:20:00 count.22:30:00 count.22:40:00
## 1               0              0              0              0
## 26              3              7              2              6
## 36              8              8              9              4
##    count.22:50:00 count.23:00:00 count.23:10:00 count.23:20:00
## 1               0              0              0              0
## 26              3              4              6             11
## 36             10             10              9              6
##    count.23:30:00 count.23:40:00 count.23:50:00 count.NA   date
## 1               0              0              0        0 990101
## 26              7              7              7        0 990102
## 36              9              9              3        0 990103
```

```
data_wide$date2 = strptime(data_wide$date, format = "%y%m%d")
data_wide$days = weekdays(data_wide$date2)  ### adding column for days of the week
data_wide$holidays = 1  ### adding a holidays column. 1 means not a holiday. 2 means a holiday. mostly i
for (i in 1:length(data_wide$date2)) {
    for (count in 1:length(holidays)) {
        if (sum(data_wide$date2[i] == holidays[count]) > 0) {
            data_wide$holidays[i] = 2
        }
    }
}
data_wide$Colour = ifelse(data_wide$days == "Friday", 1, ifelse(data_wide$days ==
    "Monday", 2, ifelse(data_wide$days == "Saturday", 3, ifelse(data_wide$days ==
    "Sunday", 4, ifelse(data_wide$days == "Thursday", 5, ifelse(data_wide$days ==
    "Tuesday", 6, 7))))))  ### adding a colour column based on day of week
```

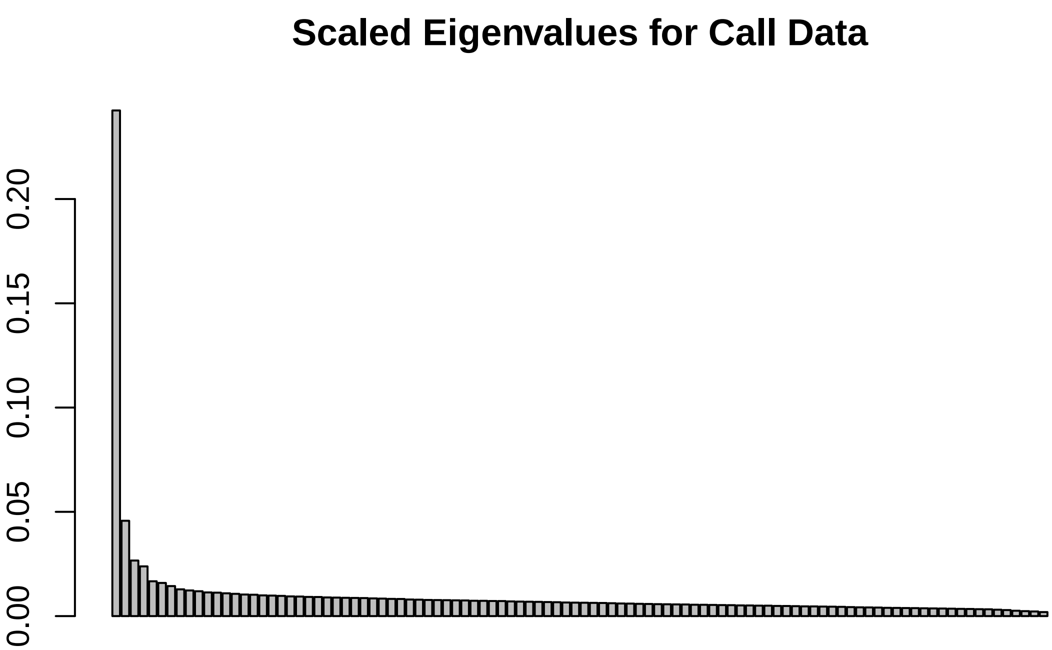
```r
#### deleting all times before 7AM when the center opens and
#### date column and convert data to matrix format
X = as.matrix(data_wide[, -c(1:29, 132:137)])
Xsvd = svd(X)
data_wide$date2 <- as.POSIXct(data_wide$date2)
data_fri = filter(data_wide, days == "Friday")
dim(data_fri)
```

```
## [1]  52 137
```

```r
data_sat = filter(data_wide, days == "Saturday")
data_rest = filter(data_wide, days != "Saturday" & days != "Friday")
X = as.matrix(data_wide[, -c(1:29, 132:137)])
X_fri = as.matrix(data_fri[, -c(1:29, 132:137)])
X_sat = as.matrix(data_sat[, -c(1:29, 132:137)])
X_rest = as.matrix(data_rest[, -c(1:29, 132:137)])
#### for poisson data apply transformation sqrt(x+1/4) to data
#### to make more normal
Y = sqrt(X + 1/4)
Y_fri = sqrt(X_fri + 1/4)
Y_sat = sqrt(X_sat + 1/4)
Y_rest = sqrt(X_rest + 1/4)
```

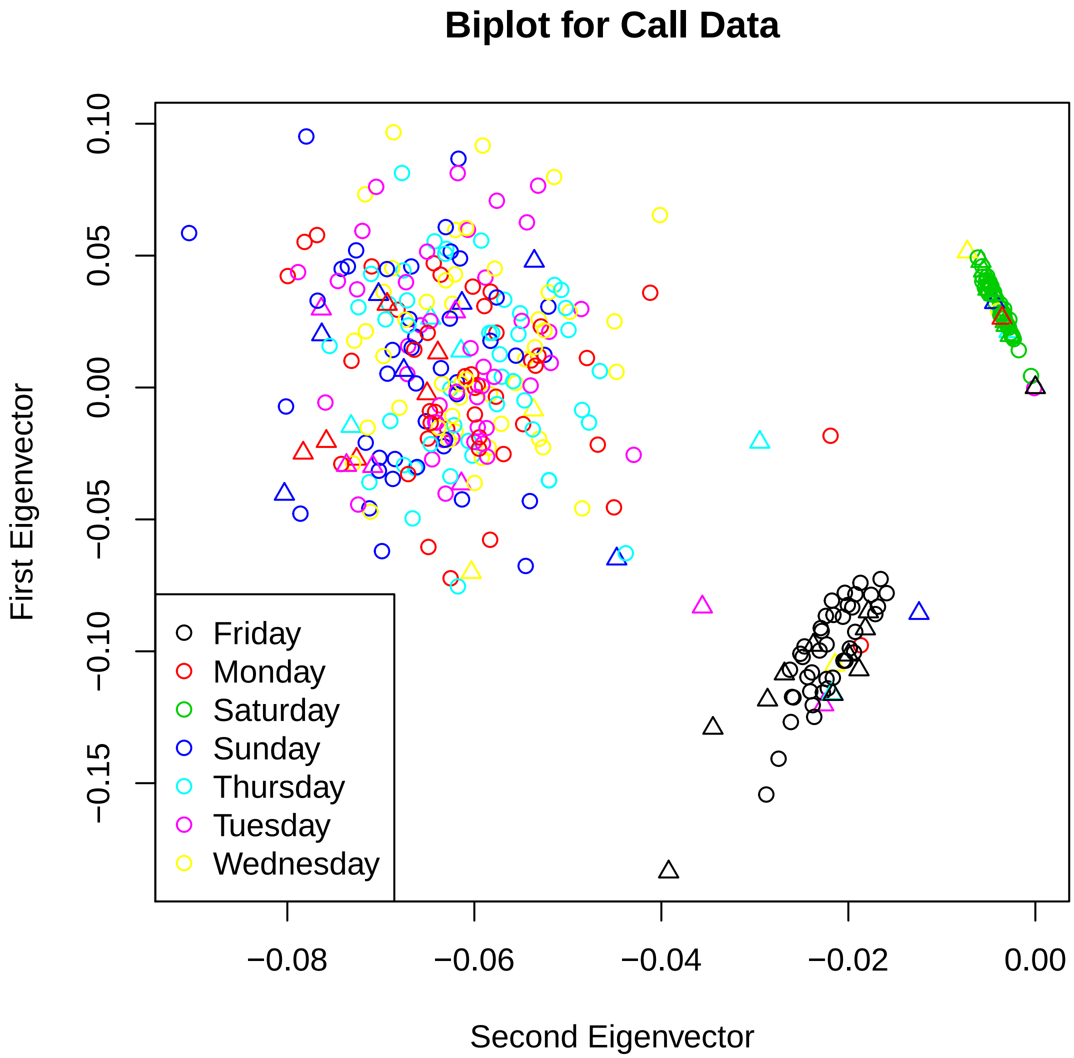
### Scree Plots

The scree plot indicates that about 25% is explained by the 1st eigenvector, about 5% by the 2nd one and so on.




### Bi-Plots

The bi-plot indicates that the weekends (Fridays and Saturdays in Israel) are separate from the rest of the data. This indicates that for the prediction phase, it might be best to work on all of these separately. The triangles indicate +/-1 day from Holidays. Almost all the outlier seem to be either the weekends or Holidays.




## Model Evaluation

To evaulate the models we use Absolute Error(AE) where

$$AE_t = \frac{1}{T}\sum_{i=T+1}^{364} |\hat{y}_{it} - y_{it}|$$

and $T$ is the size of the training dataset.

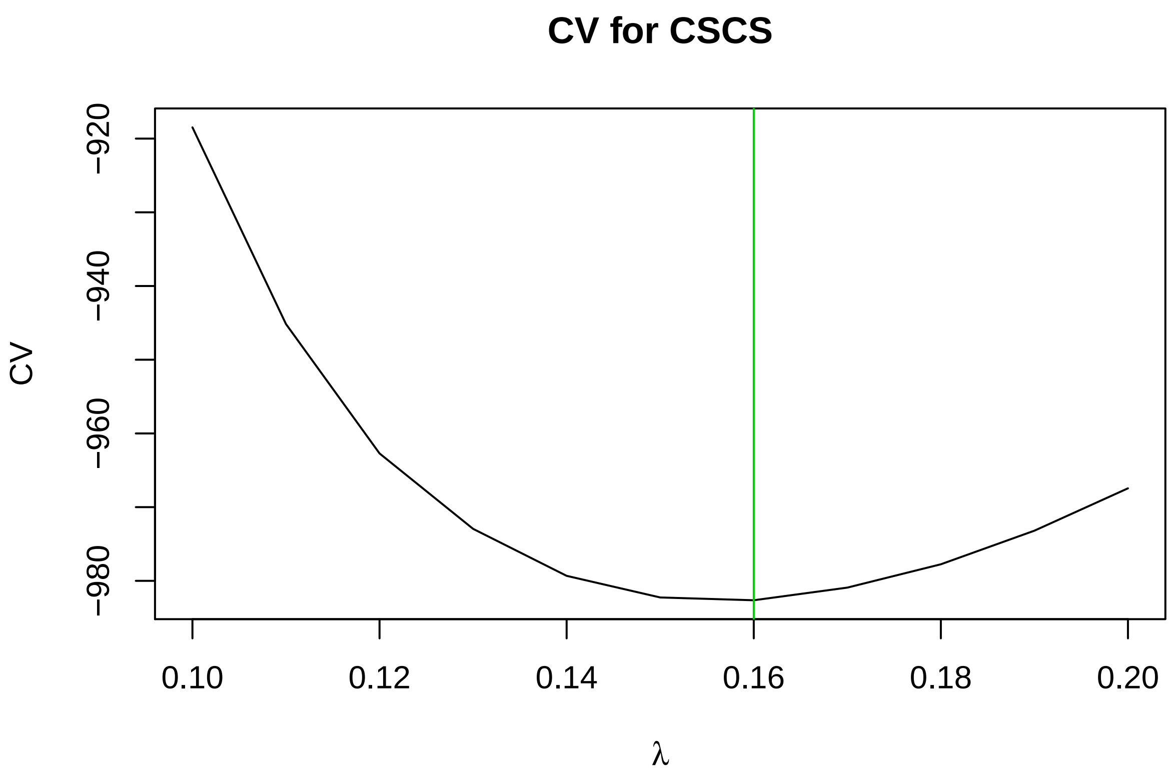
## Full dataset

Initially we do the evaluations on the full dataset (including weekends). We estimate $\mu_2$ by using the sample mean and estimate $\Sigma$ using the sample covariance method and CSCS. For CSCS, we need to pick a penalty parameter, which we do using cross-validation of the likelihood.

```
cv = function(lambda, K, train) {
    cvec = rep(0, K)
    set.seed(12345)
    index = sample(1:dim(train)[1], replace = FALSE)
    foldsize = ceiling(dim(train)[1]/K)
    inTrain <- list()
    for (i in 1:(K - 1)) {
        inTrain[[i]] <- index[((i - 1) * foldsize + 1):((i -
            1) * foldsize + foldsize)]
    }
    inTrain[[K]] <- index[((K - 1) * foldsize + 1):length(index)]
    for (k in 1:K) {
        Ytrain = train[-c(inTrain[[k]]), ]
        Ytest = train[c(inTrain[[k]]), ]
        E = Ytrain
        out = CSCS2(as.matrix(E), lambda)$L
        Sigma_v = t(out) %*% out
        s_v = dim(Ytest)[1]
        sumterm = sum(diag(Ytest %*% (Sigma_v) %*% t(Ytest)))
        cvec[k] = s_v * log(det(solve(Sigma_v))) + sumterm
    }
    cv = (1/K) * sum(cvec)
    return(cv)
}
```

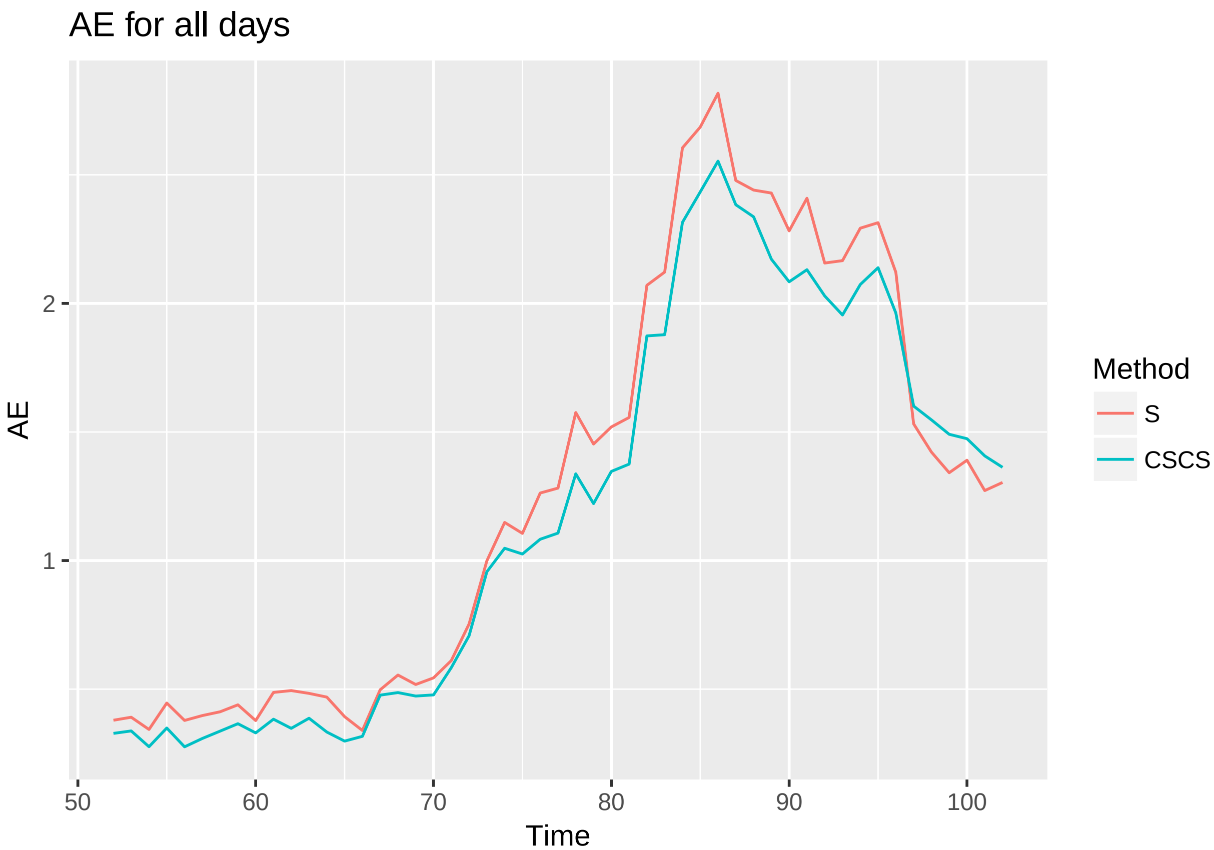
```
training = 1:252
train = scale(Y[training, ], center = TRUE, scale = FALSE)
test = Y[-training, ]
mu = colMeans(train)
S = (t(train) %*% train)/(dim(train)[1])
SError <- matrix(0, nrow = nrow(test), ncol = 51)
for (k in 1:nrow(test)) {
    S.mu2 <- predict.mean(test[k, 1:51], mu = mu, Sigma = S)
    SError[k, ] <- (S.mu2 - as.numeric(test[k, 52:102]))
}
E.S <- colMeans(abs(SError))
Time = 52:102
AE = E.S
Method = rep("S", length(AE))
Sdata = data.frame(Method, AE, Time)

lambdal = 0.01
lambda = seq(0.1, 0.2, by = lambdal)
out = rep(0, length(lambda))
for (k in 1:length(lambda)) {
    out[k] = cv(lambda[k], 5, train)
}
```



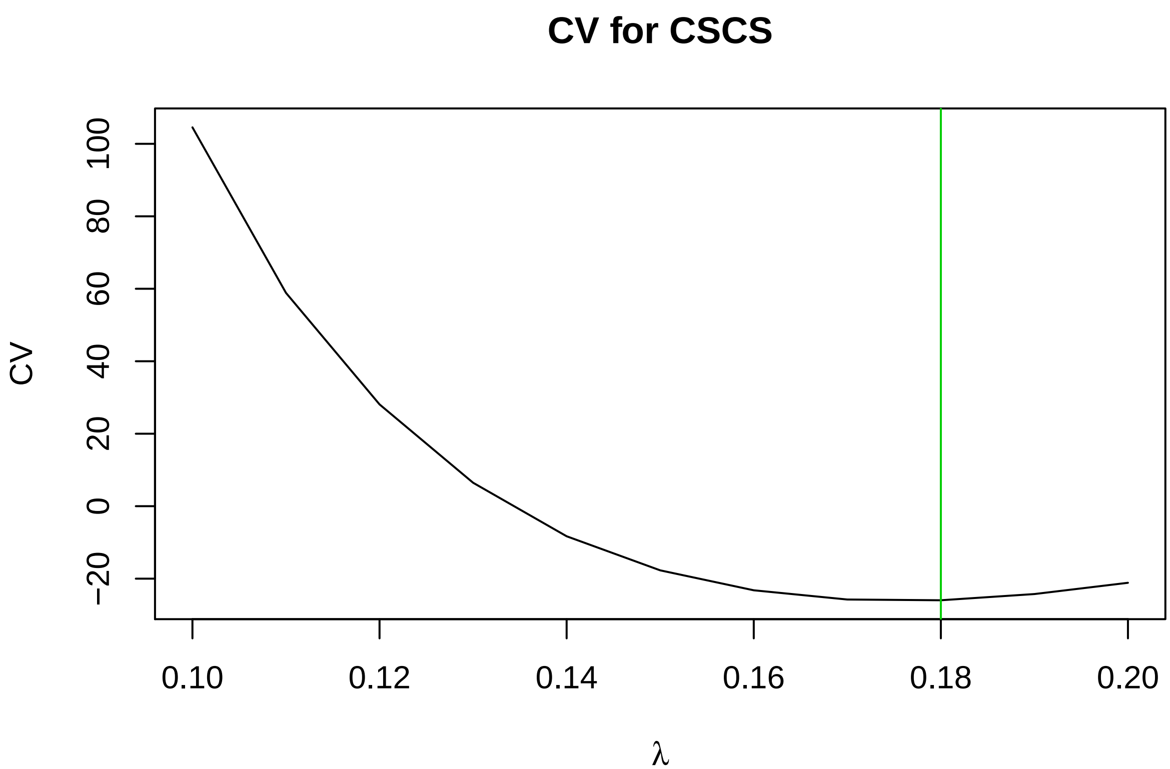

```
lambdastar = lambda[which.min(out)]
E = train
out = CSCS2(as.matrix(E), lambdastar)$L
CSCS = solve(t(out) %*% out)
CSCSError <- matrix(0, nrow = nrow(test), ncol = 51)
for (k in 1:nrow(test)) {
    CSCS.mu2 <- predict.mean(test[k, 1:51], mu = mu, Sigma = CSCS)
    CSCSError[k, ] <- (CSCS.mu2 - as.numeric(test[k, 52:102]))
}
E.CSCS <- colMeans(abs(CSCSError))
Time = 52:102
AE = E.CSCS
Method = rep("CSCS", length(AE))
CSCSdata = data.frame(Method, AE, Time)
```

The following plot shows the comparison of using the sample covariance matrix and CSCS on the full dataset. It's quite clear that CSCS does better than the sample covariance matrix in most cases.

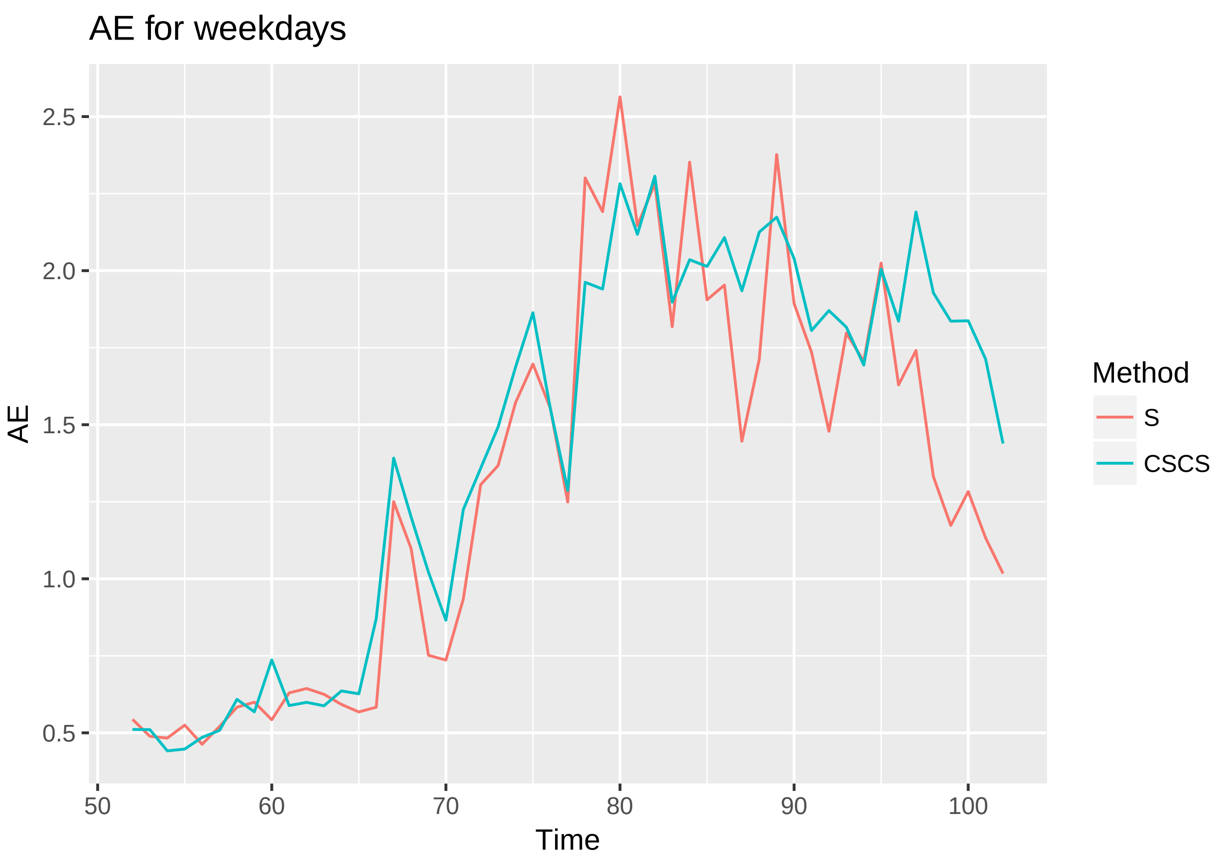



### Weekdays

This is the comparison for just the weekdays. Here, there isn't a clear winner. This is probably our dataset is more homogenous. CSCS is a more robust covariance estimator than the sample covaraince matrix. As we focussed on dataset with fewer outliers, the sample covariance matrix performed better comparitively.



CV for CSCS
CV
100
80
60
40
20
0
−20
0.10
0.12
0.14
0.16
0.18
0.20
λ

AE for weekdays
AE
2.5
2.0
1.5
1.0
0.5
50
60
70
80
90
100
Time
Method
S
CSCS

### Connections to graphical models

FInally, here is a graph showing how the time periods are related.